BASRTL

BASPOWDR LIST

```
0000      1              .TITLE   BAS$POWDR                    ; BASIC double ** float routine
0000      2              .IDENT   /1-001/                      ; File: BASPOWDR.MAR Edit:PLL1001
0000      3
0000      4      ;
0000      5      ;*************************************************************************
0000      6      ;*                                                                       *
0000      7      ;*   COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                             *
0000      8      ;*   DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.              *
0000      9      ;*   ALL RIGHTS RESERVED.                                                *
0000     10      ;*                                                                       *
0000     11      ;*   THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
0000     12      ;*   ONLY IN ACCORDANCE WITH THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
0000     13      ;*   INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER *
0000     14      ;*   COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
0000     15      ;*   OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY *
0000     16      ;*   TRANSFERRED.                                                         *
0000     17      ;*                                                                       *
0000     18      ;*   THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE *
0000     19      ;*   AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT *
0000     20      ;*   CORPORATION.                                                         *
0000     21      ;*                                                                       *
0000     22      ;*   DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS *
0000     23      ;*   SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.              *
0000     24      ;*                                                                       *
0000     25      ;*                                                                       *
0000     26      ;*************************************************************************
0000     27      ;
0000     28
0000     29      ;++
0000     30      ; FACILITY: Basic Support Library
0000     31      ;
0000     32      ; ABSTRACT:
0000     33      ;
0000     34      ;     This module contains entry points to support exponentiation
0000     35      ;     (** or ^) in BASIC-PLUS-2 for DOUBLE ** FLOAT.
0000     36      ;
0000     37      ; ENVIRONMENT: User Mode, AST Reentrant
0000     38      ;
0000     39      ;--
0000     40      ; AUTHOR: P. Levesque   , CREATION DATE: 5-Oct-81
0000     41      ;
0000     42      ; MODIFIED BY:
0000     43      ;
0000     44      ; 1-001 - Original
```

BAS$POWDR
1-001

N 10

; BASIC double ** float routine
DECLARATIONS

15-SEP-1984 23:59:02   VAX/VMS Macro V04-00   Page 2
6-SEP-1984 10:34:04   [BASRTL.SRC]BASPOWDR.MAR;1   (2)

```
        0000    46                  .SBTTL  DECLARATIONS
        0000    47  ;
        0000    48  ; INCLUDE FILES:
        0000    49  ;
        0000    50
        0000    51  ;
        0000    52  ; EXTERNAL DECLARATIONS:
        0000    53  ;
        0000    54                  .DSABL  GBL                     ; Prevent undeclared
        0000    55                                                  ; symbols from being
        0000    56                                                  ; automatically global.
        0000    57
        0000    58                  .EXTRN  OTS$POWDR               ; OTS$ double ** float exponentiation
        0000    59                  .EXTRN  OTS$POWDJ               ; OTS$ double ** int exponentation
        0000    60                  .EXTRN  BAS$K_DIVBY_ZER         ; Divide by Zero
        0000    61                  .EXTRN  BAS$K_ILLARGLOG         ; Illegal argument in LOG
        0000    62                  .EXTRN  BAS$$STOP               ; Error reporting routine
        0000    63
        0000    64  ;
        0000    65  ; MACROS:
        0000    66  ;
        0000    67
        0000    68  ;
        0000    69  ; EQUATED SYMBOLS:
        0000    70  ;
        0000    71
        0000    72  ;
        0000    73  ; OWN STORAGE:
        0000    74  ;
        0000    75
        0000    76  ;
        0000    77  ; PSECT DECLARATIONS:
        0000    78  ;
    00000000    79                  .PSECT _BAS$CODE PIC, USR, CON, REL, LCL, SHR, -
        0000    80                          EXE, RD, NOWRT, LONG
        0000    81
```

B 11

BAS$POWDR   ; BASIC double ** float routine   15-SEP-1984 23:59:02   VAX/VMS Macro V04-00   Page 3
1-001   BAS$POWDR - BASIC double ** float   6-SEP-1984 10:34:04 [BASRTL.SRC]BASPOWDR.MAR;1   (3)

```
                    0000        83                  .SBTTL  BAS$POWDR  - BASIC double ** float
                    0000        84      ;++
                    0000        85      ; FUNCTIONAL DESCRIPTION:
                    0000        86      ;
                    0000        87      ;         This routine takes BASE ** EXP, using the following table
                    0000        88      ;         for unusual cases:
                    0000        89      ;
                    0000        90      ;         BASE > 0                           Call OTS$POWDR, normal case.
                    0000        91      ;         BASE = 0, EXP > 0                  Return 0.0.
                    0000        92      ;         BASE = 0, EXP = 0                  Return 1.0.
                    0000        93      ;         BASE = 0, EXP < 0                  Error: divide by zero
                    0000        94      ;         BASE < 0, EXP even integer         Call OTS$POWDJ with -BASE
                    0000        95      ;         BASE < 0, EXP odd integer          Call OTS$POWDJ with -BASE, negate result
                    0000        96      ;         BASE < 0, EXP not integer          Error: illegal argument in LOG.
                    0000        97      ;
                    0000        98      ; CALLING SEQUENCE:
                    0000        99      ;
                    0000       100      ;         CALL result.wd.v = BAS$POWDR (base.rd.v, exponent.rf.v)
                    0000       101      ;
                    0000       102      ; INPUT PARAMETERS:
                    0000       103      ;
        00000004    0000       104      ;         base = 4
        0000000C    0000       105      ;         exponent = 12
                    0000       106      ;
                    0000       107      ; IMPLICIT INPUTS:
                    0000       108      ;
                    0000       109      ;         NONE
                    0000       110      ;
                    0000       111      ; OUTPUT PARAMETERS:
                    0000       112      ;
                    0000       113      ;         NONE
                    0000       114      ;
                    0000       115      ; IMPLICIT OUTPUTS:
                    0000       116      ;
                    0000       117      ;         NONE
                    0000       118      ;
                    0000       119      ; FUNCTION VALUE:
                    0000       120      ; COMPLETION CODES:
                    0000       121      ;
                    0000       122      ;         double result of exponentiation
                    0000       123      ;
                    0000       124      ; SIDE EFFECTS:
                    0000       125      ;
                    0000       126      ;         Will signal Divide By Zero or Illegal argument in LOG if its
                    0000       127      ;         arguments are bad, and OTS$POWDR and OTS$POWDJ may also signal.
                    0000       128      ;
                    0000       129      ;--
                    0000       130
        0000'       0000       131      BAS$POWDR::     .MASK OTS$POWDR          ; Entry point
                    0002       132                                              ; Since this routine uses no
                    0002       133                                              ; registers and usually transfers
                    0002       134                                              ; control to OTS$POWDR, we copy
                    0002       135                                              ; its register save mask and then
                    0002       136                                              ; JMP past its save mask and only
                    0002       137                                              ; save the registers once
04 AC   73          0002       138                      TSTD    base(AP)        ; Test base relationship to zero
        06   15      0005       139                      BLEQ    1$              ; If base leq 0, do case analysis
```

C 11

BAS$POWDR   ; BASIC double ** float routine   15-SEP-1984 23:59:02   VAX/VMS Macro V04-00   Page 4
1-001   BAS$POWDR - BASIC double ** float   6-SEP-1984 10:34:04 [BASRTL.SRC]BASPOWDR.MAR;1   (3)

```
        00000002'GF    17  0007   140          JMP      G^OTS$POWDR+2              ; Transfer control to the OTS$
                               000D   141                                          ; routine to do exponentiation
                               000D   142  ;+
                               000D   143  ; Come here if the base is less than or equal to zero.  We must filter
                               000D   144  ; several special cases, as described above.
                               000D   145  ;-
                    2E  13  000D   146  1$:        BEQL     4$                       ; Branch if base = 0
 50  50  08  00  0C AC  54  000F   147          EMODF    exponent(AP), #0, #1, R0, R0
                    1A  12  0016   148          BNEQ     3$                       ; Branch if exponent is not integer
                               0018   149  ;+
                               0018   150  ; The base is less than zero and the exponent is an integer.
                               0018   151  ; BASIC defines this as working the same way as if an integer was
                               0018   152  ; in the expression (making a floating variable which happens to
                               0018   153  ; contain an integer value equivalent to an integer variable).
                               0018   154  ;-
            50  0C AC  4A  0018   155          CVTFL    exponent(AP), R0          ; Convert exponent to integer
                    50  DD  001C   156          PUSHL    R0                       ; Save for even/odd test
                    50  DD  001E   157          PUSHL    R0                       ; Stack as parameter to OTS$POWDJ
         7E  04  AC  72  0020   158          MNEGD    base(AP), -(SP)           ; Stack -base also
    00000000'GF  03  FB  0024   159          CALLS    #3, G^OTS$POWDJ           ; Call integer power routines
                 03  8E  E9  002B   160          BLBC     (SP)+, 2$                 ; Branch if exponent even
            50  50  72  002E   161          MNEGD    R0, R0                   ; Exponent odd, negate the result
                    04  0031   162  2$:        RET                               ; and return with it.
                               0032   163  ;+
                               0032   164  ; Come here if the base is less than zero but the exponent is not
                               0032   165  ; an integer.  BASIC defines this as an error.
                               0032   166  ;-
         7E  00'8F  9A  0032   167  3$:        MOVZBL   #BAS$K_ILLARGLOG, -(SP)  ; Illegal Argument in LOG
    00000000'GF  01  FB  0036   168          CALLS    #1, G^BAS$$STOP          ; Never return.
                               003D   169  ;+
                               003D   170  ; Come here if the base is equal to zero.  The value we return depends
                               003D   171  ; upon the sign of the exponent.
                               003D   172  ;-
            0C AC  53  003D   173  4$:        TSTF     exponent(AP)             ; Test the exponent against zero
                 09  19  0040   174          BLSS     6$                       ; Branch if exponent lss 0
                 03  13  0042   175          BEQL     5$                       ; Branch if exponent is 0
                               0044   176  ;+
                               0044   177  ; Come here if the base is zero and the exponent is greater than zero.
                               0044   178  ; BASIC defines this as 0.0.
                               0044   179  ;-
                 50  7C  0044   180          CLRD     R0                       ; R0, R1 = 0.0
                    04  0046   181          RET                               ; Return to caller
                               0047   182  ;+
                               0047   183  ; Come here if the base is zero and the exponent is zero.  BASIC defines
                               0047   184  ; this as 1.0.
                               0047   185  ;-
            50  08  70  0047   186  5$:        MOVD     #1, R0                   ; R0, R1 = 1.0
                    04  004A   187          RET                               ; Return to caller.
                               004B   188  ;+
                               004B   189  ; Come here if the base is zero and the exponent is less than zero.
                               004B   190  ; BASIC defines this as an error.
                               004B   191  ;-
         7E  00'8F  9A  004B   192  6$:        MOVZBL   #BAS$K_DIVBY_ZER, -(SP)  ; Divide by zero
    00000000'GF  01  FB  004F   193          CALLS    #1, G^BAS$$STOP          ; Report error, never return.
                               0056   194  ;
                               0056   195          .END
```

```
BAS$$STOP           ********  X    00
BAS$K_DIVBY_ZER     ********  X    00
BAS$K_ILLARGLOG     ********  X    00
BAS$POWDR           00000000 RG    01
BASE              = 00000004
EXPONENT          = 0000000C
OTS$POWDJ           ********  X    00
OTS$POWDR           ********  X    00
```

```
                                    +-------------------+
                                    ! Psect synopsis !
                                    +-------------------+
```

PSECT name	Allocation	PSECT No.	Attributes									
. ABS .	00000000 (0.)	00 (0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE
_BAS$CODE	00000056 (86.)	01 (1.)	PIC	USR	CON	REL	LCL	SHR	EXE	RD	NOWRT	NOVEC	LONG

```
                                  +-------------------------+
                                  ! Performance indicators !
                                  +-------------------------+
```

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.07	00:00:00.40
Command processing	104	00:00:00.43	00:00:01.94
Pass 1	72	00:00:00.46	00:00:01.45
Symbol table sort	0	00:00:00.01	00:00:00.01
Pass 2	47	00:00:00.38	00:00:01.08
Symbol table output	3	00:00:00.01	00:00:00.01
Psect synopsis output	2	00:00:00.02	00:00:00.07
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	259	00:00:01.39	00:00:04.96

The working set limit was 900 pages.
2216 bytes (5 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 8 non-local and 6 local symbols.
195 source lines were read in Pass 1, producing 8 object records in Pass 2.
0 pages of virtual memory were used to define 0 macros.

```
                                  +---------------------------+
                                  ! Macro library statistics !
                                  +---------------------------+
```

Macro library name	Macros defined
_255DUA28:[SYSLIB]STARLET.MLB;2	0

0 GETS were required to define 0 macros.

There were no errors, warnings or information messages.

MACRO/ENABLE=SUPPRESSION/DISABLE=(GLOBAL,TRACEBACK)/LIS=LIS$:BASPOWDR/OBJ=OBJ$:BASPOWDR MSRC$:BASPOWDR/UPDATE=(ENH$:BASPOWDR)